An Accurate and Efficient Yield Optimization Method for Analog Circuits Based on Computing Budget Allocation and Memetic Search Technique

Bo Liu[1], Francisco V. Fernández[2], Georges Gielen[1]

[1] ESAT-MICAS, Katholieke Universiteit Leuven, Leuven, Belgium. E-mail: {Georges.Gielen, Bo.Liu}@esat.kuleuven.be

[2] IMSE, CSIC and University of Sevilla, Sevilla, Spain. E-mail: Francisco.Fernandez@imse.cnm.es

ABSTRACT

Monte-Carlo (MC) simulation is still the most commonly used technique for yield estimation of analog integrated circuits, because of its generality and accuracy. However, although some speed acceleration methods for MC simulation have been proposed, their efficiency is not high enough for MC-based yield optimization (determines optimal device sizes and optimizes yield at the same time), which requires repeated yield calculations. In this paper, a new sampling-based yield optimization approach is presented, called the Memetic Ordinal Optimization (OO)-based Hybrid Evolutionary Constrained Optimization (MOHECO) algorithm, which significantly enhances the efficiency for yield optimization while maintaining the high accuracy and generality of MC simulation. By proposing a two-stage estimation flow and introducing the OO technology in the first stage, sufficient samples are allocated to promising solutions, and repeated MC simulations of non-critical solutions are avoided. By the proposed memetic search operators, the convergence speed of the algorithm can considerably be enhanced. With the same accuracy, the resulting MOHECO algorithm can achieve yield optimization by approximately 7 times less computational effort compared to a state-of-the-art MC-based algorithm integrating the acceptance sampling (AS) plus the Latin-hypercube sampling (LHS) techniques. Experiments and comparisons in 0.35 μm and 90nm CMOS technologies show that MOHECO presents important advantages in terms of accuracy and efficiency.

Keywords

Yield optimization, variation-aware analog sizing, process variation, OO, Monte-Carlo, memetic algorithm

1. INTRODUCTION

Industrial analog integrated circuit design not only calls for fully optimized nominal design solutions, but also requires high robustness and yield in the light of varying supply voltage and temperature conditions, as well as inter-die and intra-die process variations [1,2]. The flow and available methods of yield optimization are summarized in Fig. 1. The yield analysis can be classified into statistical methods and non-statistical methods. Non-statistical methods include corner-based methods [2] and performance-specific worst-case design (PSWCD) methods [3]. Such methods are efficient due to the limited number of simulations. But their drawback is the difficulty to know in advance the worst-case corner points, and it may result in serious design overkill [1]. Moreover, simple and fast sensitivity analysis in worst-case design methods may harm the accuracy in nanometer technologies. Statistical methods include Monte-Carlo (MC)-based methods and response-surface-based (RSB) methods [1,4]. The latter category often has to face dilemmas considering the balance between the accuracy and the complexity of the model, as well as the accuracy and the number of samples. Hence, non-statistical and RSB methods can be used for yield optimization but have significant accuracy problems. MC-based methods have the advantages of generality and high accuracy [5], so they are still the most reliable and commonly used technique for yield estimation till now. Nevertheless, a large number of simulations are needed for MC analysis, therefore limiting its use within an iterative yield optimization loop, as in Fig. 1. Although some speed enhancement techniques for MC simulation have been proposed [6,7], the efficiency gain is not high enough to make MC-based yield optimization practical. Therefore, in this paper we address yield optimization by proposing a different (but complementary) approach: we dramatically increase the efficiency of yield optimization by optimally allocating the computing budget to candidate solutions and by enhancing the convergence speed of the search strategy by means of a memetic algorithm in combination with the state-of-the-art sampling techniques.

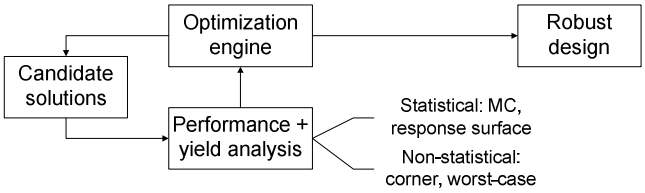

Fig.1. Review of existing yield optimization methods

Based on the above ideas, we propose the memetic Ordinal Optimization (OO)-based hybrid evolutionary constrained optimization algorithm (MOHECO). The method tries to:

- avoid over-design;
- be general enough to be applied to any analog circuit in any technology and any distribution of the process parameters;
- simultaneously handle inter-die and intra-die variations in nanometer technologies;
- provide high-accuracy results comparable to MC;
- consume far less computational effort compared with state-of-the-art MC-based methods.

The remainder of the paper is organized as follows. Section 2 introduces the basic components and the general framework of MOHECO. Section 3 tests MOHECO by practical examples. The concluding remarks are presented in Section 4.

2. THE MOHECO ALGORITHM

2.1 Available Speed Enhancement Techniques for MC-based Yield Estimation

To satisfy the first four goals, MC analysis is selected. Two speed enhancement techniques for MC-based yield optimization are available. The first one is the acceptance sampling (AS) technique [6]. The method only samples near the border of the acceptance region, instead of sampling over the whole space during the MC

analysis. Note that the original AS method is RSB (regression)-based. In order to guarantee the accuracy, the AS technique implemented in this paper samples the border of the acceptance region by MC simulations. The second technique consists in using a Design of Experiment (DOE) technique like Latin Hypercube Sampling (LHS) [7] to replace Primitive Monte Carlo (PMC) simulation [8]. These improvements are important, but from our experiments, it is shown that the computational load is still too large for yield optimization in real practice.

2.2 Basics of MOHECO

Most of the current analog circuit sizing and yield optimization methodologies are evolutionary computation (EC)-based, which rely on the evolution of a population of candidate solutions [9]. Owing to this, the computational effort at each iteration and the necessary number of iterations are two key problems that affect the speed of the yield optimization. We solve these two problems by optimally allocating the computing budget to each candidate in the population (reduce computational effort at each iteration) and by improving the search mechanism (decrease necessary number of iterations). Therefore, the computational effort can be further reduced considerably. To the best of our knowledge, these two ideas have not been incorporated before in yield optimization.

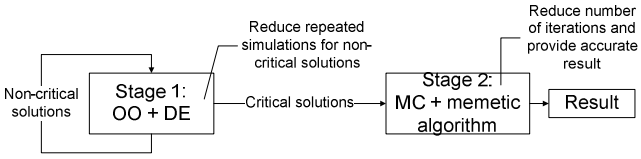

Fig. 2. Two-stage memetic optimization flow

Our idea is shown in Fig. 2. In order to optimally allocate the computing budget at each iteration, instead of assigning the same number of simulations for the MC simulations of all solutions selected by AS in the whole process, the yield optimization process is divided in two stages. In the first stage, the ranking of the candidate solutions and a reasonably accurate yield estimation result for good (critical) solutions are important. For medium or bad (non-critical) candidate solutions, their ranking is highly important, but an accurate yield estimation is not significant. The reason is that the function of the yield estimations for non-critical candidates is to guide the selection operator in the EC algorithm, but the candidates themselves are not selected as the final result or even to enter the second stage. Hence, the computational efforts spent on feasible but non-optimal candidate solutions can be strongly reduced. On the other hand, the estimations to non-critical candidates cannot be too inaccurate. After all, correct selection of candidate solutions in the yield optimization algorithm is necessary. In this stage, the yield optimization problem is formulated as an ordinal optimization problem, targeted at identifying critical candidate solutions, allocating enough number of samples to the MC simulation of these solutions, while reasonably few samples are allocated to non-critical solutions [10].

In the second stage, an accurate result is highly important, so the number of simulations in each yield estimation increases in this stage to obtain an accurate final result. However, this is not as cheap as yield analysis in the first stage. We focus on decreasing the necessary number of iterations in the second stage. Instead of using conventional EC algorithms, e.g. genetic algorithm, as the search engine, we propose a memetic algorithm. We use the differential evolution (DE) algorithm [11] (a powerful and fast global optimization algorithm) for global search and the Nelder-Mead (NM) simplex method [12] for local search. Moreover, we combine them in a special way fit for yield optimization. In the following, the basic components of MOHECO will be introduced first, and the general framework will then be presented.

2.3 Introducing Ordinal Optimization into Yield Optimization

Ordinal optimization (OO) has emerged as an efficient technique for simulation and optimization, especially for problems where the computation of the simulation models is time-consuming [10,13]. OO is based on two basic tenets: (a) "order" is easier than "value" (ordinal optimization converges exponentially with stochastic simulation models whereas the convergence rate of cardinal optimization is $1/\sqrt{n}$); and (b) an accurate estimation is very costly but a good enough design can be obtained much easily. Therefore, OO fits quite well the objectives of the first stage of MOHECO. In the first stage, a bunch of good designs are selected and sent to the second stage. The requirement is correct selection with a reasonably accurate estimation and with the smallest computational effort. Consider the yield evaluation function $J(\hat{x},\hat{\xi})$, where $\hat{\xi}$ represents the process variations and $\hat{x}$ represents each of the S candidate solutions $x_1, x_2, \cdots x_s$. For a single simulation, we define $J(\hat{x},\hat{\xi})$ =1 if all the circuit specifications are met, and $J(\hat{x},\hat{\xi})$ =0 otherwise. Because the MC simulation determines the yield as the ratio of the functional chips to all fabricated chips, the mean value of $J(\hat{x},\hat{\xi})$, $\overline{J}$, corresponds to the yield value. Let us consider a total computing budget equal to T simulations. In yield optimization, T is affected by the number of feasible solutions at each generation. Here, we set T to $sim_{ave} \times N_{fea}$, where N_{fea} is the number of solutions selected by AS and sim_{ave} is the average budget for each candidate. The budget allocation problem consists in determining the number of simulations $n_1, n_2, \cdots, n_s$ of the S candidate solutions such that $n_1 + n_2 + \cdots n_s = T$. An asymptotic solution to the optimal computing budget allocation problem is proposed in [13]:

$$n_b = \sigma_b \left(\sum_{i=1, i\neq b}^{S} n_i^2 / \sigma_i^2\right)^{1/2} \tag{1}$$

$$n_i / n_j = \left(\frac{\sigma_i / \delta_{b,i}}{\sigma_j / \delta_{b,j}}\right)^2, \quad i, j \in \{1, 2, \cdots, S\}, \ i \neq j \neq b$$

where b is the best design of the S candidate solutions (represented by the best mean value), $\sigma_1^2, \sigma_2^2, \cdots, \sigma_S^2$ are the finite variances of the S solutions, respectively, and $\delta_{b,i} = \overline{J}_b - \overline{J}_i$ represents the deviations of the mean values with respect to the best design.

A typical population in example 1 (section 3) is selected as an illustrative example to show the benefits of OO (see Fig. 3): candidates with a yield value larger than 70% correspond to 36% of the population, and are assigned 55% of the simulations. Candidates with a yield value smaller than 40% correspond to 30% of the population, and are only assigned 13% of the

simulations. The total number of simulations is 11% compared with that of the MC plus AS and LHS method.

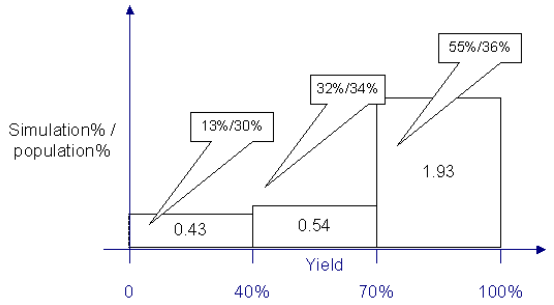

Fig. 3. The function of OO in one typical population

Candidates with estimated yield larger than a threshold value (we set candidates with yield>97% in the first stage) enter the second stage. In this stage of MOHECO, all the candidates are assigned the upper limit in number of samples to guarantee the accuracy of the final result; while other candidates in the population still remain in the first stage and still use the evolution method described previously. Note that the two stages are not a division on time, but a division on different yield estimation methods.

2.4 Construction of the Memetic Search Engine

Apart from introducing OO to decrease the computational effort in one iteration in the first stage, decreasing the necessary number of iterations is another key problem. The DE algorithm [11] is selected as the global search engine. DE uses a simple differential operator to create new candidate solutions and a one-to-one competition scheme to greedily select new candidates. The DE algorithm outperforms many EC algorithms in terms of solution quality and convergence speed [9,11]. More details are in [11]. Although DE is a very powerful and fast global optimization algorithm, it is not so efficient in local tuning to reach the exact optimal solution (other global optimization algorithms, e.g. genetic algorithm also have the same problem). But local tuning corresponds to the second stage of MOHECO, where each candidate has the upper limit of simulations, which is expensive. As decreasing the number of iterations is a critical problem, we propose a memetic algorithm. Memetic algorithms [15] use, besides the global optimization engine, a population-based strategy coupled with individual search heuristics capable of performing local refinements. Here, we use the DE algorithm as the global optimizer (exploration) to obtain the sub-optimal solution and the Nelder-Mead (NM) simplex method [12] as the local search engine to refine the result provided by DE (exploitation). The NM simplex algorithm is a direct search method for multi-dimensional unconstrained optimization. This method does not need numerical or analytical gradients.

Unlike the normal construction of a memetic algorithm, it is not wise to add the NM simplex method to all the candidates of the DE population in the second stage. The NM simplex method needs about 10 iterations for one candidate and each iteration needs $n_{\max}$ simulations. Multiplied by the size of the population, this is also an expensive procedure. We propose a method of only doing local search to the best member of the DE population. The best member in the population of DE is used to generate all the candidates in the next iteration, whose useful schemata will be shared by all the generated candidates. Another point is that the local search is not necessary in each iteration. We only trigger it when the yield value cannot be improved by the DE operators for 5 iterations. If this occurs, we use the memetic operators to search near the best member of the current population and then come back to DE. By this method, the memetic search method can be used in an expensive MC-based optimization loop. Experimental results show that this method enhances the convergence speed considerably.

Besides the two key ideas described above, we use the selection-based method [16] to handle the constraints caused by the circuit performance specifications. Its advantages and its combination with DE algorithm for analog sizing have been shown in [9]. In addition, the AS and DOE techniques are also integrated for separate candidates.

2.1 The General Framework of MOHECO

Based on the above components, the MOHECO algorithm can be constructed. The detailed flow diagram is shown in Fig. 4.

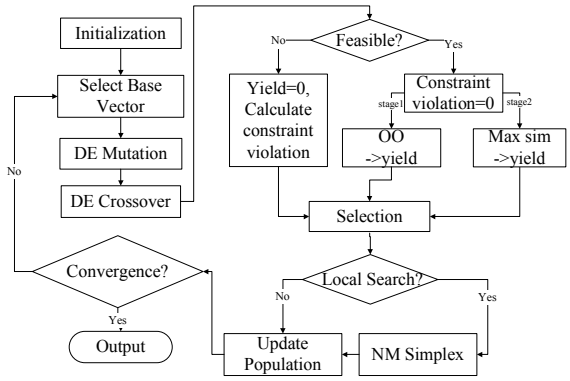

Fig.4. Flow diagram of MOHECO

The algorithm consists of the following steps.

Step 0: Initialize the parameters n_0, T, Δ, $n_{\max}$, the sampling method (e.g. PMC, LHS) in OO, and the algorithm parameters (e.g. the population size). Initialize the population by randomly selecting values of the design variables within the allowed ranges.

Step 1: Select the current best candidate using the criterion described in section 2.4.

Step 2: Perform the DE mutation and crossover operation to obtain each individual's trial individual.

Step 3: Check the feasibility of the candidates. For feasible solutions, go to step 4; for infeasible solutions, go to step 7.

Step 4: Set constraint violations equal to 0, and check the candidate solution belongs to stage 1 or stage 2. If belongs to stage 1, go to step 5, otherwise, go to step 6.

Step 5: Use the OO technique described in section 2.3 to calculate the yield.

Step 6: Use maximum simulation number to calculate the yield.

Step 7: Set yield equal to 0, and calculate constraint violations. No yield is estimated in this step.

Step 8: Perform selection between each individual and its corresponding trial counterpart according to the selection rule described in section 2.4.

Step 9: Check if the condition of using local search operators is

met. If yes, go to step 10; otherwise go to step 11.

Step 10: Perform NM simplex search described in section 2.4.

Step 11: If the stopping criterion is met (e.g. a convergence criterion or a maximum number of generations), then output X_{best} and its objective value; otherwise go back to Step 1.

3. Experimental Results and Comparisons

In this section, the MOHECO algorithm is demonstrated by two practical analog circuits in 0.35 μm and 90nm CMOS technologies. The key techniques in this paper, i.e. the OO for yield optimization and the memetic search engine, are emphasized here. The comparisons between PSWCD method and RBS methods are carried out in section 3.4. Because the advantages of the DE algorithm compared with other EC algorithms in analog sizing have been demonstrated in [9], such comparisons will not be shown here. In all methods, the AS and LHS technique are used with MC simulation in all experiments. All experiments also use the DE optimization engine and the selection-based constraint handling mechanism. The population size is 50, the crossover rate is 0.8 and the DE step size is 0.8. The optimization stops when the reported yield reaches 100%, or when the yield does not increase for 20 subsequent generations. Parameter n_0 is set to 15 and sim_{ave} is set to 35 in all the experiments. These two parameters have been determined based on experiments in different circuits under different conditions. The examples have been run on a PC with 4GB RAM and Linux operating system, in the MATLAB environment. Synopsys's HSPICE is used as the circuit performance evaluator.

3.1 Experimental Method

There is not much sense in comparing the efficiency of different methods without a good accuracy. The accuracy is determined by the number of samples used for MC analysis of each feasible solution. For example, two experiments using 50 or 500 simulations for each feasible candidate can report a solution with "100% yield". But obviously the sampling error of using 50 samples is larger than using 500 samples. Our approach is to first perform the yield optimization by applying the AS and the DOE methods with different numbers of simulations to each candidate, and then calculate their accuracy. We calculate the difference between the reported yield and that estimated by 50,000 simulations. The results by 50,000 simulations are shown to stay below 0.01% compared to the difference obtained with 250,000 simulations (using different random numbers and different candidates), which is a very reliable approximation of the real yield value. After that, we select an appropriate one, which can be defined as the one that meets the designer's requirements of accuracy and calls for a good balance on efficiency. We then compare the AS plus LHS / PMC method to MOHECO.

Another problem is that the performance of evolutionary algorithms (EA) may be affected by the random numbers used in the evolution operators. Therefore, 10 runs with independent random numbers have been performed for all experiments and the results have been analyzed and compared statistically.

3.2 Example 1

The MOHECO algorithm is first tested on a fully differential folded-cascode amplifier, shown in Fig. 5, implemented in a 0.35 μm CMOS process with 3.3V power supply. The specifications are $A_0 \geq 70\text{dB}$, $GBW \geq 40\text{MHz}$, $PM \geq 60^\circ$, $OS \geq 4.6V$, $power \leq 1.07mW$ (The reason to choose 1.07mW is that according to experiments, 1.08mW is easy to meet, but 1.06mW cannot reach 100% yield). The total number of the process variation variables is 80, including 15 transistors $\times$ 4 (TOX,VTH0, LD,WD) = 60 intra-die variables (mismatch) and 20 inter-die variables (TOXRn,VTH0Rn, DELUON, DELL, DELW, DELRDIFFN, VTH0Rp, DELUOP, DELRDIFFP, CJSWRn, CJSWRp, CJRn, CJRp, NPEAKn, NPEAKp, TOXRp, LDn, WDn, LDp, WDp). Statistical information of the process variables has been extracted from the information provided by the foundry.

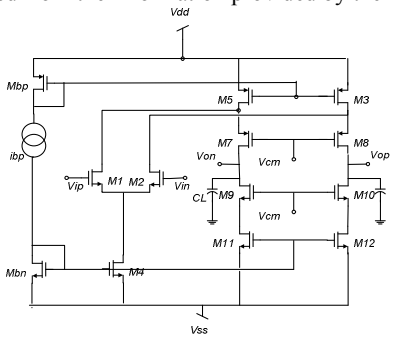

Fig.5. Fully differential folded-cascode amplifier

Experiments with 300, 500 and 700 simulations to each feasible candidate by the AS plus LHS method have been done. In this first example we separately study the improvement provided by the introduced OO technique and that provided by the memetic search engine. The deviation to the yield estimate provided by a 50,000 MC simulation analysis at the same design point and the total number of simulations is analyzed. The statistical results of 10 independent runs are shown in Table 1, Table 2 and Fig. 6.

Table 1. Deviation of the yield results by different methods from the yield estimated by a 50000 MC simulation analysis for example 1

methods	best	worst	average	variance
300 simulations (AS + LHS)	0.22%	1.94%	0.78%	2.5e-5
500 simulations (AS + LHS)	0.01%	0.65%	0.32%	4.5e-6
700 simulations (AS + LHS)	0	0.74%	0.26%	5.5e-6
OO+AS+LHS	0.02%	0.76%	0.47%	7.5e-6
MOHECO	0.04%	0.63%	0.32%	3.6e-6

From Table 1, it can be concluded that the accuracy with 300 simulations to each candidate is too low: the mean deviation is almost 0.8%, whereas the deviation reduces to about 0.3% when using 500 or 700 simulations. The worst-run comparisons also show that the error is nearly 2% if we use 300 simulations. From Fig. 6, 500 simulations seems a good choice: it can provide a relatively accurate result in a reasonable time for this circuit. Using 700 simulations for the MC simulation of each candidate can provide a little more accurate result, but the computational effort is nearly 2.5 times that of using 500 simulations. Thus, 500 simulations for each feasible candidate is chosen. Then, 10 experiments of MOHECO and the technique of OO plus AS plus LHS are performed.

Table 2. Total number of simulations for example 1

methods	best	worst	average	variance
300 simulations (AS + LHS)	38300	301200	113480	5.4e+5
500 simulation (AS + LHS)s	43500	337500	186350	1.1e+6
700 simulations (AS + LHS)	221200	736400	447790	2.9e+6
OO+AS+LHS	16531	76048	43151	4.1e+8
MOHECO	14012	43053	26208	1.0e+8

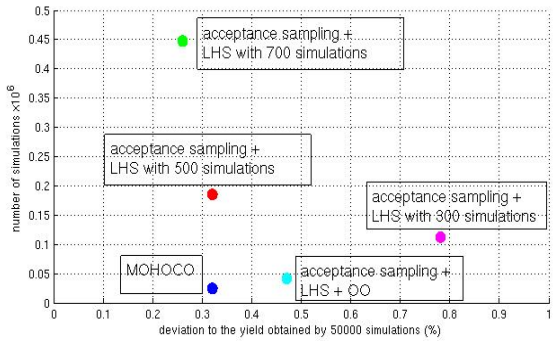

Fig. 6. Comparisons of average yield estimate deviation and number of simulations for different methods for example 1

From Fig.6, we can see that the deviations of MOHECO from the accurate yield estimate are very close to that of using 500 simulations and the computational cost are much lower. The deviations obtained by OO plus AS plus LHS are a little larger, but also acceptable. With respect to the number of simulations, shown in Table 2, MOHECO costs only 1/7 (14.06%) of the simulations of the AS plus LHS-based method with comparable accuracy. This is the contribution of both the OO and memetic search techniques. Without the memetic operators, as can be seen from the result of the OO plus AS plus LHS method, it spends 1/4.3 (23.16%) of the simulations of AS plus LHS-based method. The comparisons are shown in Fig. 6 more clearly. The average time cost of MOHECO for this example is about 5 minutes.

3.3 Example 2

The second example is a two-stage telescopic cascode amplifier, shown in Fig. 7, in a 90nm CMOS process with 1.2V power supply. The specifications are $A_0 \geq 60\text{dB}$, $GBW \geq 300\text{MHz}$, $PM \geq 60^\circ$, $OS \geq 1.8V$, $power \leq 10mW$, $\sqrt{area} \leq 180\mu\text{m}$ and $offset \leq 0.05mV$. Like the previous example, all the transistors must be in the saturation region. The total number of process variation variables for this technology is 123, including 19 transistors$\times$ 4(TOX,VTH0, LD,WD) = 76 intra-die variables and 47 inter-die variables. The degree of difficulty of yield optimization is directly affected by the severity of the specifications. Even without considering the process variations (the goal is to satisfy all the specifications), these specifications are also very challenging. We have tried the memetic single objective evolutionary algorithm (MSOEA) for high-performance analog sizing [9], selection-based differential evolution (SBDE) [9], genetic algorithm and differential evolution plus penalty function. Only MSOEA and SBDE succeeded, but at a cost of 200 to 300 generations with a population of 60 candidates. In contrast, if not considering process variations, the first example has converged in 20-30 generations. Therefore, we use this example to test the ability of MOHECO under extremely severe performance constraints. The experimental method and parameter settings are the same as in example 1.

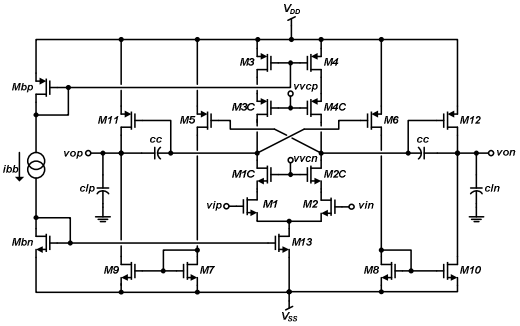

Fig.7. Fully differential two-stage telescopic cascode amplifier

Table 3. Deviation of the yield results by different methods from the yield estimated by a 50000 MC simulation analysis for example 2

methods	best	worst	average	variance
300 simulations (AS + LHS)	0.6%	2.05%	1.20%	2.4e-5
500 simulations (AS + LHS)	0.7%	1.32%	0.89%	4.2e-6
MOHECO	0.12%	0.85%	0.52%	4.1e-6

Table 4. Total number of simulations for example 2

methods	best	worst	average	variance
300 simulations (AS + LHS)	195000	3384200	1600170	1.1e+12
500 simulations (AS + LHS)	92000	7349000	3444850	5.9e+12
MOHECO	96632	1185495	487946	6.8e+11

From Table 3, it can be seen that using 500 simulations to each feasible solution can result in a yield value comparable to MOHECO. From Table 4, it can be seen that MOHECO costs only 14.16% of the number of simulations of the AS plus LHS-based method, while giving a more accurate result. It can be concluded that MOHECO outperforms the other methods in efficiency while keeping similar or even slightly better accuracy. On the other hand, as shown in Table 4, although the number of simulations of the other methods reaches 10^6, the number of simulations of MOHECO also reaches 10^5, which may cost a few hours in real practice.

3.4 Comparisons to Other Approaches

Then, the comparisons with the PSWCD methods and RSB methods are discussed. In PSWCD methods, although over-design is strongly reduced compared with corner-based methods, over-design still exists, which eliminates good designs. The reason is that for each candidate design point, each specification is considered as a separate inner optimization problem to obtain its worst case. But the corresponding decision variables (process variation parameters) for different specifications are different. In another word, the separated worst-case points cannot be achieved simultaneously, so their combination is over-estimated.

In RSB methods, the yield of a circuit is formulated as a black-box model of the design parameters and the process variation parameters. The data obtained from expensive MC simulations at a number of design points is used to generate a model able to predict the yield in other design points much cheaper than with a MC simulation at the price of a loss of accuracy. To assess this loss of accuracy we will consider a response-surface method based on neural networks (NN), often considered as a powerful regressor [17]. Here, we will use a Backward Propagation NN [17] with 20 neurons in the hidden layer and the Levenberg-Marquardt algorithm for training it to approximate the yield. The basic philosophy behind a RSB method, is to use as few expensive training MC simulations as possible and, comparatively, as many cheap yield predictions as possible. However, an acceptable accuracy of the yield predictions is needed, and accuracy can only be improved by increasing the number of training data. Therefore, there is a trade-off between accuracy and training data (hence computation time) for which a quantitative analysis is convenient. For this goal, we will use the data generated during a typical execution of MOHECO in example 1. We will consider the data generated up to a given iteration of MOHECO as training data, and use the data of subsequent iterations to assess the accuracy of the black-box model.

The MOHECO typical case converges to 100% yield in 51 iterations. At every iteration, we use the data from all previous iterations to train the NN and use this to predict the yield values of the current iteration. The error between the predicted yield values and the real yield values obtained by MC simulations is then calculated. We checked that even when the training data corresponding to the first 50 iterations of MOHECO are used, the RMS error is still 6.86%. So the black-box model would be of little use as a much more accurate method (MOHECO) is available for the same computational effort. Therefore, for nanometer technologies, response-surface-based methods can hardly achieve an acceptable accuracy for a reasonably low training time.

4. CONCLUSIONS

In this paper, the MOHECO algorithm has been proposed for yield optimization of analog integrated circuits, considering both the inter-die and intra-die process variations. The method is general and has no risk of over-design. MOHECO can provide high-accuracy results with far less computational costs (about 1/7) even when compared with the state-of-the-art MC-based methods. The reason is that: (1) it uses a two-stage yield optimization process and OO in the first stage, which determines the simulation effort for each candidate solution "intelligently"; (2) a memetic algorithm combing the DE and the NM simplex algorithm is used and triggered by an adaptive rule enhance the convergence speed of the search engine considerably; (3) the state-of-the-art AS and DOE techniques and the selection-based constraint-handling technique also contribute positively to MOHECO. Therefore, MOHECO is an efficient good candidate for analog circuit yield optimization, especially for new nanometer technologies with large variability.

5. ACKNOWLEDGMENTS

This research was supported by a special bilateral agreement scholarship of Katholieke Universiteit Leuven, Belgium and Tsinghua University, P. R. China, and by the TIC-2532 Project, funded by Consejeria de Innovación, Ciencia y Empresa, Junta de Andalucia, Spain. We sincerely thank Dr. Brecht Machiels, ESAT-MICAS, K. U. Leuven, for valuable discussions.

6. REFERENCES

[1] G. Gielen, et al., 2007. "Automated Synthesis of Complex Analog Circuits", *Proc. of 18th European Conf. on Circuit Theory and Design*, pp. 20-23.

[2] K. S. Eshbaugh, 1992. "Generation of Correlated Parameters for Statistical Circuit Simulation", *IEEE TCAD*, pp. 1198-1206.

[3] F. Schenkel, et al., 2001. "Mismatch Analysis and Direct Yield Optimization by Spec-Wise Linearization and Feasibility-Guided Search", *Proc. of DAC*, pp. 858-863.

[4] S. Basu, et al., 2009. "Variation-Aware Macromodeling and Synthesis of Analog Circuits using Spline Center and Range Method and Dynamically Reduced Design Space", *Proc. of 22nd International Conf. on VLSI Design*, pp. 433-438.

[5] A. Mutlu, et al., 2003. "Concurrent Optimization of Process Dependent Variations in Different Circuit Performance Measures", *Proc. of the 2003 International Symposium on Circuits and Systems*, pp. 692-695.

[6] N. Elias, 1994. "Acceptance Sampling: An Efficient, Accurate Method for Estimating and Optimizing Parametric Yield [IC Manufacture]" , *IEEE Journal of Solid-State Circuits*, pp. 323-327.

[7] M. Stein, 1987. "Large Sample Properties of Simulations Using Latin Hypercube Sampling," *Technometrics*, pp. 143-151.

[8] S. Tiwary, et al., 2006. "Generation of Yield-Aware Pareto Surfaces for Hierarchical Circuit Design Space Exploration", *Proc. of DAC*, pp. 31-36.

[9] B. Liu, et al., 2009. "A Memetic Approach to the Automatic Design of High Performance Analog Integrated Circuits", *ACM Trans. on Design Automation of Electronics Systems*, 14(3), Article 42.

[10] Y. Ho, et al., Ordinal optimization. Soft optimization for hard problems. Springer, 2007.

[11] K. Price, et al., 2005. Differential Evolution. A Practical Approach to Global Optimization. Springer, Berlin, Heidelberg, New York.

[12] J. Lagarias, et al., 1998. "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions", *SIAM Journal of Optimization* 9(1), pp. 112-147.

[13] C.H. Chen, et al., 2000. "Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization", *Discrete Event Dynamic Systems: Theory and Applications*, pp. 251-270.

[14] C. Chen, et al., 2000. Computing efforts allocation for ordinal optimization and discrete event simulation. *IEEE Trans. on Automatic Control*, Vol. 45, No. 5, pp. 960-964.

[15] P. Moscato, 1989. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. *Technical Report 158–79*, Caltech Concurrent Computation Program, California Institute of Technology.

[16] K. Deb, 2000. "An Efficient Constraint Handling Method for Genetic Algorithm", *Computer Methods in Applied Mechanics and Engineering*, pp. 311-338.

[17] PD Wasserman, 1988. Neural computing: theory and practice. New York: Van Nostrand Reinhold.